

Programmable Storage

Carlos Maltzahn, UC Santa Cruz

10/6/2020

Snowmass Community Planning Meeting

Opportunities for computing R&D to advance particle physics

Carlos Maltzahn

people.ucsc.edu/carlosm



A screenshot of a web browser displaying the profile page of Carlos Maltzahn at UC Santa Cruz. The browser's address bar shows the URL https://users.soe.ucsc.edu/~carlosm. The page features a navigation bar with links to Home, News, Research, Publications, Teaching, People, Contact, and CV. The main content area includes a circular profile picture of Carlos Maltzahn, his name, and his title as Adjunct Professor, Founder & Director of CROSS. It also lists his affiliation with the Department of Computer Science & Engineering, the Center for Research in Open Source Software (CROSS), and the Jack Baskin School of Engineering at UC Santa Cruz. Social media icons for email, calendar, home, GitHub, LinkedIn, and Twitter are shown, along with a CV icon. The page is divided into sections: Introduction, Administrative Staff, Research Staff, Current Ph.D. Students, and Graduated Ph.D. Students. The Introduction section provides a detailed biography of Dr. Maltzahn, mentioning his work on CROSS, the Systems Research Lab, and his academic background. The Administrative Staff section lists Stephanie Lieggi and Lavinia Preston. The Research Staff section lists Kate Compton and Ivo Jimenez. The Current Ph.D. Students section lists Saheed Adepoju, Xiaowei Chu, Jianshen Liu, Esmail Mirvakili, and Yiming Zhang. The Graduated Ph.D. Students section lists Alexander Ames, Joe Buck, Adam Crume, Latchesar Ionkov, Ivo Jimenez, Michael Sevilla, Andrew Shewmaker, Dimitrios Skouritis, and Noah M. Watkins, each with their thesis year and title.

Computational Storage: History

- Idea dates back to mainframes
 - First Channel I/O processors in IBM 709, 1957
- Network Attached Secure Disks (NASD)
 - Research project at CMU, 1997-2001
 - Encryption, compression, data management (“active storage”)
 - SCSI T10 Object Storage Device (OSD) v1 and v2 standards
 - Only offloads part of file system functionality
- Ceph
 - Research project at UC Santa Cruz, 2005-2007
 - Designed for OSDs
 - Broke OSD standard with P2P communication for failure management
 - Implemented for hosts, not devices
- SkyhookDM Plugin for Ceph
 - CROSS incubator project at UC Santa Cruz since 2016
 - Offloads data management of tabular data
 - Turns Ceph into an Apache Arrow-native store (since 2020)
- Computational Storage
 - SNIA Technical Working Group (TWG) since 2019
 - Focus on storage devices
- Eusocial Storage Devices
 - CROSS research project at UC Santa Cruz since 2017
 - P2P communication, specialization into “castes”
 - I/O stack flexible about offloading: pushdown, pushback
 - Leverages Smart NICs

The screenshot shows a web browser displaying the SNIA Educational Library page. The page has a purple header with navigation links: ABOUT, STANDARDS, EDUCATION, TECHNOLOGY FOCUS AREAS, NEWS & EVENTS, RESOURCES, and MEMBERSHIP. The main content area is titled "What Is Computational Storage?" and includes a definition: "Computational Storage is defined as architectures that provide Computational Storage Services coupled to storage, offloading host processing, or reducing data movement. A Computational Storage Service (CSS) is a data service or information service that performs computation on data where the service and data are associated with a storage device." Below the text are two diagrams: "Current Compute / Storage Architecture" and "Computational Storage Architecture". The "Current Compute / Storage Architecture" diagram shows a CPU connected to a storage device via a network. The "Computational Storage Architecture" diagram shows a CPU connected to a storage device via a network, with a red arrow pointing to the storage device labeled "API definitions here". The page also features a sidebar with a list of links under the heading "What Is...?".

Home » Education » What Is...? » What Is Computational Storage?

What Is Computational Storage?

Computational Storage is defined as architectures that provide Computational Storage Services coupled to storage, offloading host processing, or reducing data movement. A Computational Storage Service (CSS) is a data service or information service that performs computation on data where the service and data are associated with a storage device.

Current Compute / Storage Architecture

Computational Storage Architecture
Move Compute Closer To Storage

Computational Storage architectures enable improvements in application performance and/or infrastructure efficiency through the integration of compute resources: directly with storage, near the storage or between the host and the storage. These compute resources are outside of the traditional compute and memory architecture.

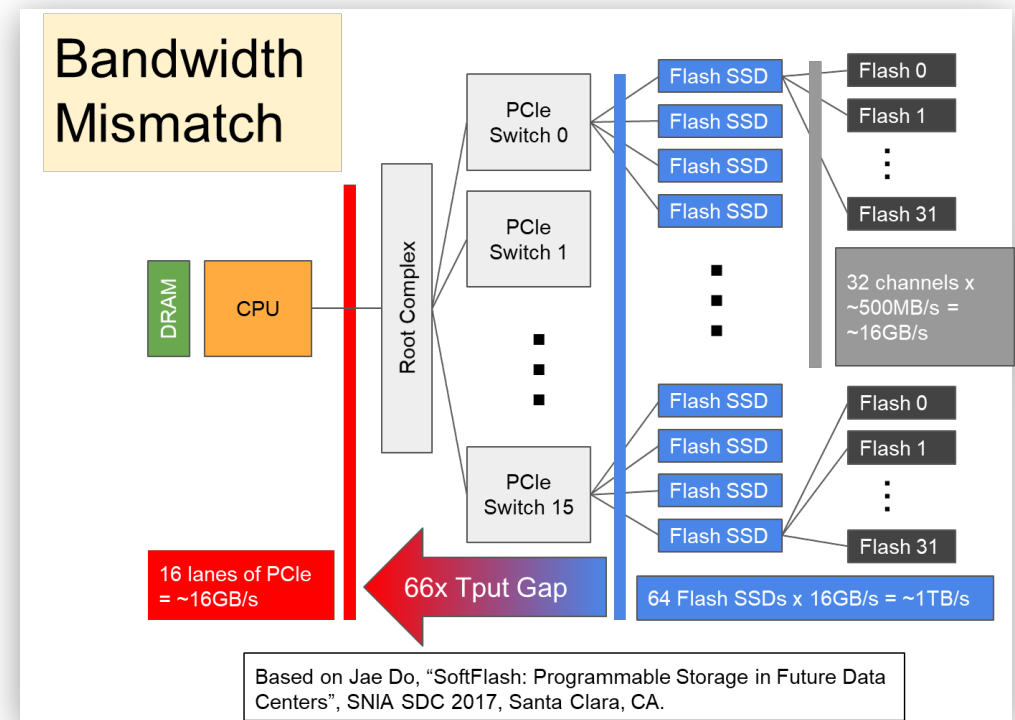
The goal of these architectures is to: enable parallel computation; reduce I/O traffic; and/or to alleviate other constraints on existing compute, memory, storage, and I/O.

The SNIA Physical Storage Technology Focus Area includes information on Computational Storage Technical Work Group activities. SNIA's Educational Library includes the SNIA Dictionary, webcasts, videos, and presentations on Computational Storage.

Learn more about Computational Storage in our Ed... Chat now

Computational Storage: Why now?

- Storage devices are getting very fast
 - CPU/DRAM/PCIe cannot keep up
 - CPU/DRAM/PCIe tax for storage increases
- Disaggregation in data centers
 - Multi-tenant workloads are too diverse for any kind of packaging
 - Better to dynamically assemble systems from parts
- Storage fabrics are expensive
 - NVMe requires host kernel resources
 - Ethernet is much cheaper and keeps getting faster
 - New IP protocols are getting very fast: e.g. HTTP/3



Programmable Storage

A programmable storage system or device exposes internal subsystem abstractions as “interfaces” to enable the creation of higher-level services via composition.

Collaborators: Jeff LeFevre (UCSC), Ivo Jimenez (UCSC), Esmail Mirvakili (UCSC), Jayjeet Chakraborty (NIT), Aditi Gupta (NIT), Aaron Chu (UCSC), Xiongfeng Song (Rice)

Programmable Storage

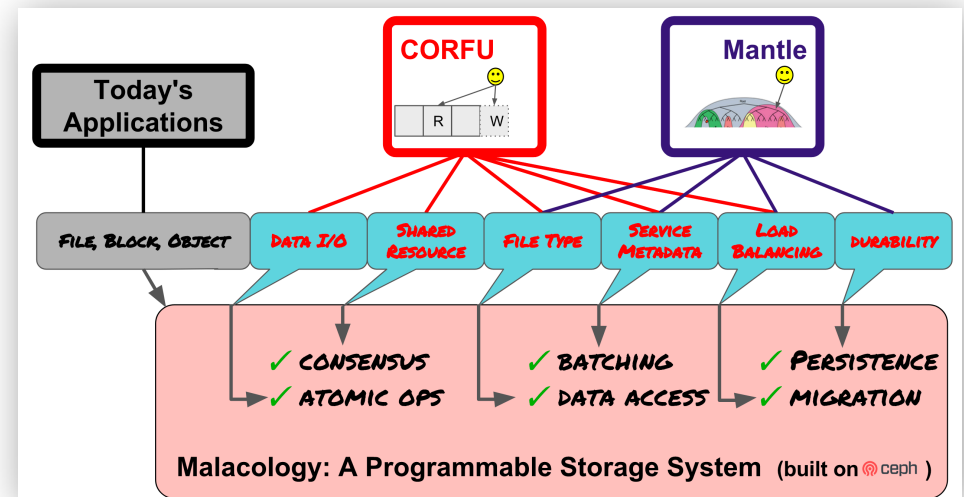
Computational Storage + Programmability

For storage systems:

- Storage has to be correct, otherwise data loss
- Correct software takes time
- Reuse as much as possible → Composability
- Composability important for optimization

For storage devices:

- Storage device industry has very low margins
- Products must fit existing market and have a minimum life time
- Programmable devices to reduce market risk
- Greater opportunity for innovation

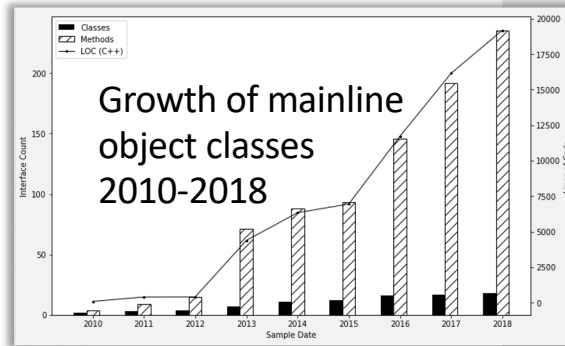


What is SkyhookDM?



```
tree -d  
ceph/src
```

```
├── cls  
│   ├── cephfs  
│   ├── hello  
│   ├── journal  
│   ├── lock  
│   ├── log  
│   ├── lua  
│   ├── numops  
│   ├── rbd  
│   ├── refcount  
│   ├── replica_log  
│   ├── rgw  
│   ├── sdk  
│   ├── statelog  
│   └── tabular  
├── timeindex  
├── user  
└── version
```



SkyhookDM

An object “class” for Ceph

- No upstream modifications required
- Inherits Ceph’s properties now and in the future
- Can use all other object extensions
- **Not a database**

Storing *tabular* data in objects

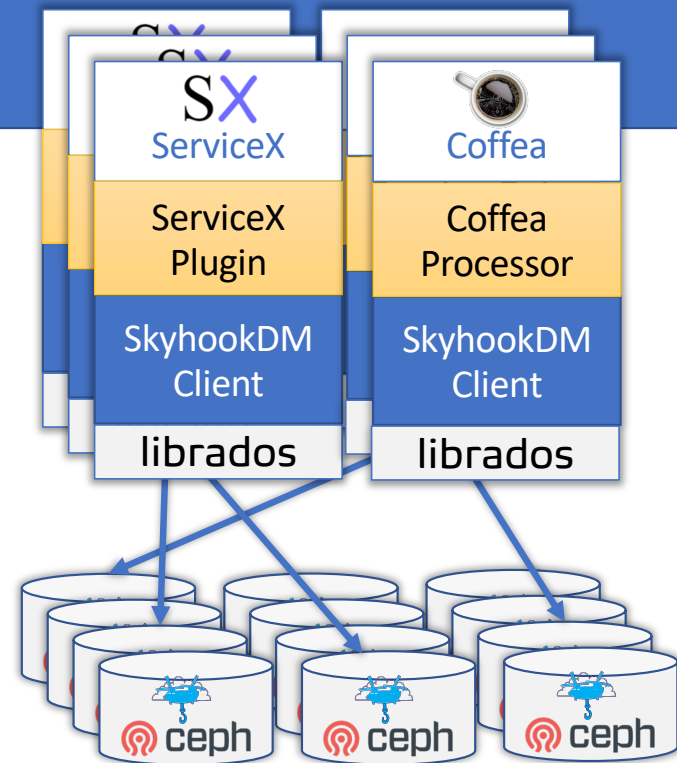
- Using 

Object read/write operations

- Select, Project, Aggregate
- Create, append rows/columns
- Indexing
- Intra- & inter-object transformations

SkyhookDM Client

- Client maps *tables* to *sets of objects*
 - Map is also stored in objects
- Client API designed for *plugins*
 - Allows *pushdown* to *scale out* tabular data operations
 - Reduces data movement (CPU cycles!)
- IRIS-HEP
 - Connting to Coffea and ServiceX
- CROSS
 - Plugins for Postgres, Spark, Pandas, HDF5



How does SkyhookDM fit into DOMA?

ServiceX Plugin and Coffea Processor:

ServiceX creates one table per *transformation request*

- Partitions table and assigns transformer to each partition
- Each transformer creates and writes an object row-by-row
- ServiceX provides table metadata, incl. partitioning to SkyhookDM

Table names are arbitrary strings, globally unique

- Column names are arbitrary strings, unique within table
- Rows have a key, unique within table

Coffea Processor:

SkyhookDM can create views across arbitrary sets of tables

- View names are arbitrary strings, globally unique
- Views can be either by reference or by copy (i.e. materialized)
- SkyhookDM stitches views on a best-effort basis
- Key mappers can map one kind of key to another and can be stored

Design allows evolution of higher-level automation.

- Table naming conventions might indicate compatibility to other tables
- View naming conventions might allow automatic reuse of materialized views
- Column naming conventions might allow versioning
- Naming convention might allow automatic garbage collection

